**Holidays: A Database for Automations**

**Holidays: A Database for Automations**

Dylan Johnson

Department of Computer Science, Concordia University

CSC 230: Database Design

Professor Ivan Gappy

September 26th, 2022

Automations have become essential to a lot of business practices. Even though programs may not be fully autonomous, there are still plenty of programs that will create, process, and even deploy data. These tools have allowed employees and developers alike to have more time to spend on more business-oriented work, rather than busy work that is very easy for a computer to process. The next step in these processes would be to take human interaction completely out of the picture. There are plenty of tasks that require human interaction to get started and even tell the program what to do as it works through itself. Looking past some particulars of specific programs, there are programs that need to be ran every business day. The challenge there is, not every Monday to Friday is a business day (and depending on the business need, the weekends as well). There have been plenty of attempts to work around this specific need, however one that could be very helpful is to have a database that stores information on days that could and would affect business practices.

**Industry Overview**

A database of this caliber could be implemented at any number of companies. If created as an open-source project any business could adapt it to their specific needs. Without getting too deep into the possibilities of a fully functioning product, the idea would be to store and make accessible to any and all programs, so that the automations will only run when they are needed, or when information is available. This would allow them to make fully autonomous solutions for any number of business needs and practices.

**Issue/Opportunity**

A lot of current programs need to be kicked off by an employee or have a trigger that will make the program run. This requires a lot of human interaction. Even with all the experience in the world, humans are still prone to errors, some are better at this than others. By having this database, any task could be automated from start to finish, taking away the need for human interaction, with the exception of quality control. A lot of programs are run after the initial process has been kicked off. The idea here is to have pre-set times any day that will start a process. If the day is not a workday, and does not need to be run, this database will tell the program that, which will cancel the process. If the program requires information from previous days, this database will tell the program that the direct previous day is not the day to work with. A computer is only as smart as it is programmed, therefore it needs to have all the information possible to complete a task correctly and efficiently, the first time.

**Database Solution**

This would be a very simple database, that would require some upkeep as the days in question are not the same year over year. While the design of this database is not advanced by any means, the possibilities it will allow for is endless. The information this database will store could start out with a relatively small amount of information that would grow as more opportunities arise for ease of use to the developer that is attaching their specific program to it. With the idea that this would be an ever-growing database, there are some basic principles that would need to be included to be useful out of the box. Firstly, this database is essentially a calendar storing information on what holidays are on a specific day. Once every Holiday is outlined for each day, it can then be labeled as what kind of Holiday it is. Since not every Holiday has an effect on business practices, there would be a column labeling it that day with keywords to identify if this program is causing a disruption to business flow or not. At its most basic components, that is all it would need with current use cases in mind.

**Database Normalization Process**

When working through the normalization process there were multiple instances of the initial database design that did not pass the test. As a result, there has been a complete rework of the design, one that would yield the same result, but in a more functional/realistic way. The database was broken up into more functional tables that can provide more accurate output providing more efficient results. The initial design was essentially one big table with each column built on the previous. That does not pass second normal form. To fix this each aspect that was not dependent on the primary key was broken off into its own table. This allows for a user to query the database and join data as needed. Instead of the database being the desired output, the database provides otherwise useless data which can be made useful through a query.

**Implementation Process**

Implementing this database will be fairly straightforward. Since this is a minimalistic design that does not need to hold a high level of data, it will not require a lot of space or processing power. Any user that is accessing this database would not require any special permissions as the database is, at its core, a calendar. The users accessing this database would start out with developers, using it as an on/off switch for the automations they have created. Once those have been implemented, the developer would not need to look at it again. The third-party program would talk to the database. The program is essentially asking “is today a workday” to which the database will respond with either “yes” or “no”. The applications tapping into this database could be any range of applications that can run automations. Applications that would otherwise run every day of the week, would still run every day, however, they would be stopped before implementing anything if there is no reason to run that given day.

**Conclusion**

The original intention of this database is to ease the production of automations. Instead of having to hard code any and every use case into a given program that would have to be changed every year. This database would store all that information in a central location. Then, any program would be able to call this database to know if it should be running, or if it can take a day off. Therefore, allowing for a one-time connection for any application to this database. The developer will never have to worry again if their program is completing tasks that it should not be on a given day.

Table 1 ERD:

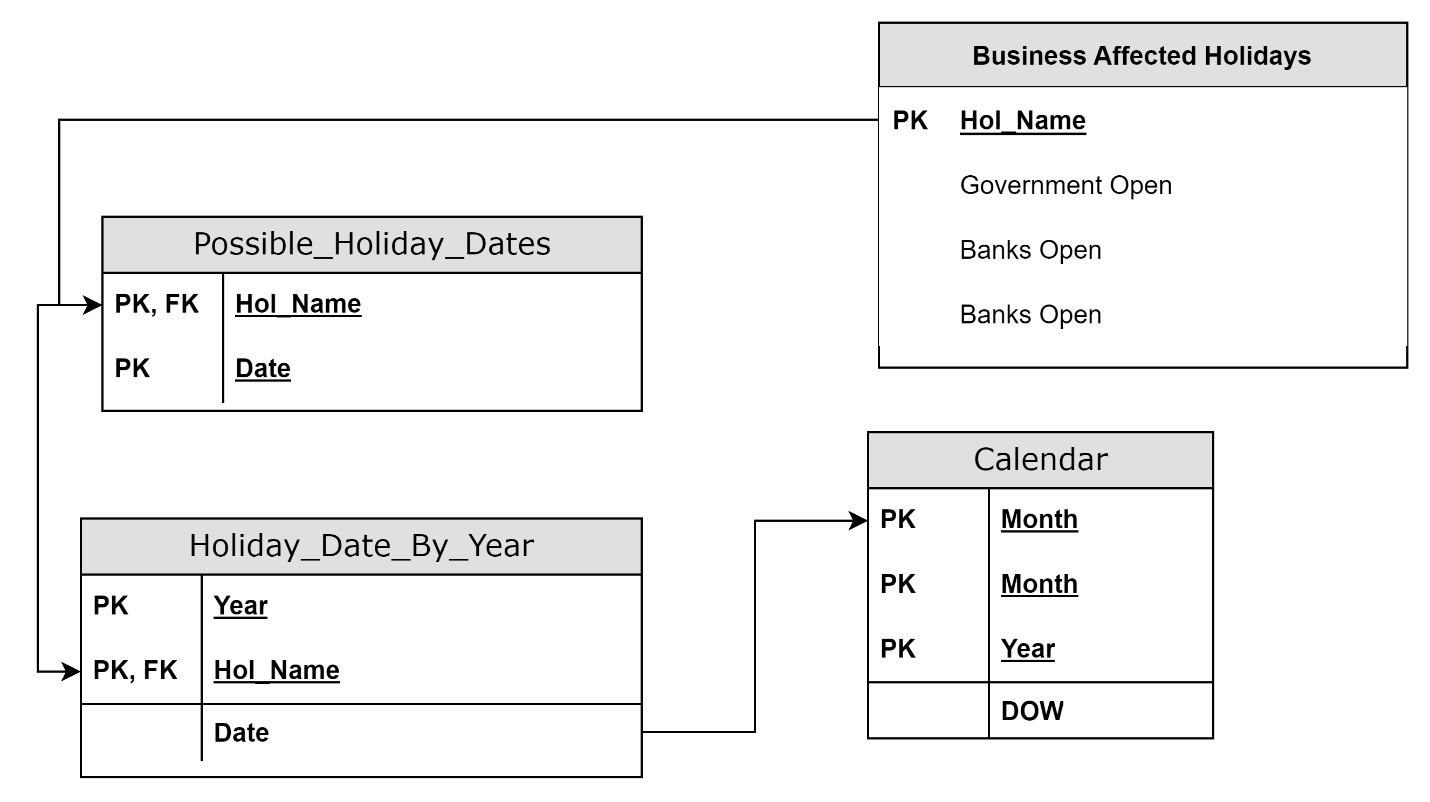


Table 2 Data Dictionary:

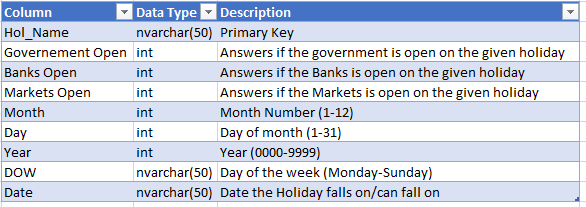
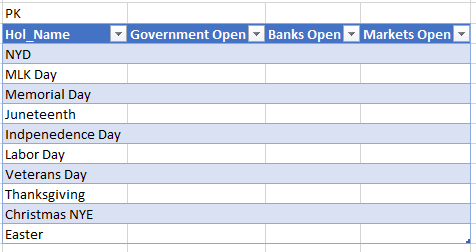
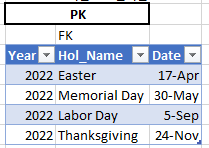
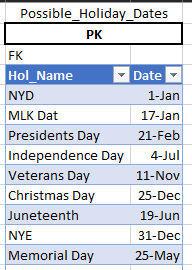
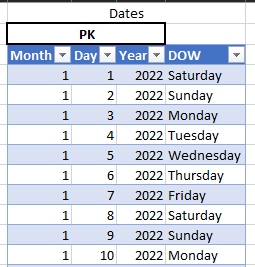


Table 3 Database Tables:





**Personal Reflection**

Through the process of creating this database project I learned a lot. Having worked with databases a lot over the years I knew they were a lot of work, but I never actually knew what all that work was. The sheer amount of validation steps to go through to make sure a database is not redundant or easily broken is, again, something I had though of but did not know that there were steps that already existed. If I could go back and start this project over again, it for sure would have taken way less time to figure all of that out. Making an affective design that is easily implanted into existing systems is no small feat but one that is well worth the work. As far as submitting this to an organization. Along the way of putting this together I discovered that there are already many iterations of this same idea available on the internet. So, if I were to propose, it would not be to create it, but to implement one that already exists.